Express Mail Label No. E[illegible]7532136 US

CLAIMS

What is claimed is:

1. In a system for producing dynamically generated content in response to user content requests that generate database queries, a method for invalidating dynamically generated content stored in caches of network devices, comprising:

receiving updates to a database of the system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing is performed using a delayed version of said database that reflects the state of the database at the beginning of a preceding processing cycle, and an update log that reflects all updates since the beginning of the preceding processing cycle.

2. The method claimed in claim 1, wherein said processing comprises:

determining whether respective results sets for subsets of said set of queries are empty or non-empty for tuples added to and deleted from said database between the beginning of a preceding processing cycle and the beginning of the current processing cycle; and

for each non-empty results set, designating the queries of the corresponding subset as ones of said identified queries.

3. The method claimed in claim 2, further comprising, if a results set for a given query Q is non-empty, designating as identified queries one or more additional queries that are dependent from said query Q in a positive query lattice.

4. The method claimed in claim 2, further comprising, if a results set for a given query Q is non-empty, skipping processing of one or more additional queries that are dependent from said query Q in a negative query lattice.

5. The method claimed in claim 1, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

6. The method claimed in claim 1, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

7. The method claimed in claim 1, wherein a subset of said queries comprises queries represented by a common query type and including a join operation, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, and to relations of said database utilized by queries of said query type for which there have been no updates since the preceding invalidation cycle, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

8. In a system for producing dynamically generated content in response to user content requests that generate database queries, a method for invalidating dynamically generated content stored in caches of network devices, comprising:

receiving updates to a database of the system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing comprises, for a given query Q,

determining whether a results set is empty or non-empty, the results set consisting of the results:

$$[(R^{+}_{1} \bowtie R_{new,2} \bowtie \ldots \bowtie R_{new,n}) \cup \ldots \cup (R_{new,1} \bowtie \ldots \bowtie R^{+}_{i} \bowtie \ldots \bowtie R_{new,n}) \cup \ldots \cup (R_{new,1} \bowtie \ldots \bowtie R_{new,n-1} \bowtie R^{+}_{n})] \cup$$
$$[(R^{-}_{1} \bowtie R_{old,2} \bowtie \ldots \bowtie R_{old,n}) \cup \ldots \cup (R_{old,1} \bowtie \ldots \bowtie R^{-}_{i} \bowtie \ldots \bowtie R_{old,n}) \cup \ldots \cup (R_{old,1} \bowtie \ldots \bowtie R_{old,n-1} \bowtie R^{-}_{n})]$$

where each R_n is a relation involved in the query Q, each $R_{old,n}$ is a state of the relation R_n as of the beginning of a preceding processing cycle, each $R_{new,n}$ is a state of the relation R_n as of the beginning of the current processing cycle, each R^{-}_{n} is a subset of tuples deleted from the relation R_n since the beginning of a preceding processing cycle, and each R^{+}_{n} is a subset of tuples added to the relation R_n since the beginning of a preceding processing cycle, and

if said results set is non-empty, designating said query Q as one of said identified queries.

9. A system for producing dynamically generated content in response to user content requests that generate database queries, comprising:

a database management system;

a web server for receiving user requests for dynamically generated content and for providing said dynamically generated content; and an application server coupled between the database management system and the web server for generating database polling queries corresponding to said requests and for supplying resulting data from the database to said web server,

the system performing an invalidation method to invalidate dynamically generated content generated by said system and stored in caches of network devices, said method comprising:

receiving updates to the database management system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing is performed using a delayed version of said database that reflects the state of the database at the beginning of a preceding processing cycle, and an update log that reflects all updates since the beginning of the preceding processing cycle.

10. The system claimed in claim 9, wherein said processing comprises:

determining whether respective results sets for subsets of said set of queries are empty or non-empty for tuples added to and deleted from said database between the beginning of a preceding processing cycle and the beginning of the current processing cycle; and

for each non-empty results set, designating the queries of the corresponding subset as ones of said identified queries.

11. The system claimed in claim 10, said processing further comprising, if a results set for a given query Q is non-empty, designating as identified queries one or more additional queries that are dependent from said query Q in a positive query lattice.

12. The system claimed in claim 10, said processing further comprising, if a results set for a given query Q is non-empty, skipping

processing of one or more additional queries that are dependent from said query Q in a negative query lattice.

13. The system claimed in claim 9, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

14. The system claimed in claim 9, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

15. The system claimed in claim 9, wherein a subset of said queries comprises queries represented by a common query type and including a join operation, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, and to relations of said database utilized by queries of said query type for which there have been no updates since the preceding invalidation cycle, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

16. A system for producing dynamically generated content in response to user content requests that generate database queries, comprising:

a database management system;

a web server for receiving user requests for dynamically generated content and for providing said dynamically generated content; and

an application server coupled between the database management system and the web server for generating database polling queries corresponding to said requests and for supplying resulting data from the database to said web server,

the system performing an invalidation method to invalidate dynamically generated content generated by said system and stored in caches of network devices, said method comprising:

receiving updates to the database management system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing comprises, for a given query Q,

determining whether a results set is empty or non-empty, the results set consisting of the results:

$$[(R^{+}_{1} \bowtie R_{new,2} \bowtie \ldots \bowtie R_{new,n}) \cup \ldots \cup (R_{new,1} \bowtie \ldots \bowtie R^{+}_{j} \bowtie \ldots \bowtie R_{new,n}) \cup \ldots \cup (R_{new,1} \bowtie \ldots \bowtie R_{new,n-1} \bowtie R^{+}_{n})] \cup$$
$$[(R^{-}_{1} \bowtie R_{old,2} \bowtie \ldots \bowtie R_{old,n}) \cup \ldots \cup (R_{old,1} \bowtie \ldots \bowtie R^{-}_{j} \bowtie \ldots \bowtie R_{old,n}) \cup \ldots \cup (R_{old,1} \bowtie \ldots \bowtie R_{old,n-1} \bowtie R^{-}_{n})]$$

where each R_n is a relation involved in the query Q, each $R_{old,n}$ is a state of the relation R_n as of the beginning of a preceding processing cycle, each $R_{new,n}$ is a state of the relation R_n as of the beginning of the current processing cycle, each R^{-}_{n} is a subset of tuples deleted from the relation R_n since the beginning of a preceding processing cycle, and each R^{+}_{n} is a subset of tuples added to the relation R_n since the beginning of a preceding processing cycle, and

if said results set is non-empty, designating said query Q as one of said identified queries.

17. In a system for producing dynamically generated content in response to user content requests that generate database queries, a method for invalidating dynamically generated content stored in caches of network devices, comprising:

receiving updates to a database of the system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing is performed using said database, which is prevented from being updated during said processing so as to reflect the state of the database at the beginning of said processing, and an update log that reflects all updates to the database since the beginning of the preceding processing cycle.

18. The method claimed in claim 17, wherein said processing comprises:

determining whether respective results sets for subsets of said set of queries are empty or non-empty for tuples added to and deleted from said database between the beginning of a preceding processing cycle and the beginning of the current processing cycle, using said locked database and said update log of said database; and

for each non-empty results set, designating the queries of the corresponding subset as ones of said identified queries.

19. The method claimed in claim 18, further comprising, if a results set for a given query Q is non-empty, designating as identified queries one or more additional queries that are dependent from said query Q in a positive query lattice.

20. The method claimed in claim 18, further comprising, if a results set for a given query Q is non-empty, skipping processing of one or more additional queries that are dependent from said query Q in a negative query lattice.

21. The method claimed in claim 17, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

22. The method claimed in claim 17, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

23. The method claimed in claim 17, wherein a subset of said queries comprises queries represented by a common query type and including a join operation, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, and to relations of said database utilized by queries of said query type for which there have been no updates since the preceding invalidation cycle, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

24. In a system for producing dynamically generated content in response to user content requests that generate database queries, a method for invalidating dynamically generated content stored in caches of network devices, comprising:

receiving updates to a database of the system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing comprises, for a given query Q,

determining whether a results set is empty or non-empty, the results set consisting of the results:

$[(R^{+}_{1} \bowtie R_{new,2} \bowtie \ldots \bowtie R_{new,n}) \cup \ldots \cup (R_{new,1} \bowtie \ldots \bowtie R^{+}_{j} \bowtie \ldots \bowtie R_{new,n}) \cup \ldots \cup (R_{new,1} \bowtie \ldots \bowtie R_{new,n-1} \bowtie R^{+}_{n})] \cup$

$[(R^{-}_{1} \bowtie (R_{new,2} \cup R^{-}_{2}) \bowtie \ldots \bowtie (R_{new,1} \cup R^{-}_{1})) \cup \ldots \cup ((R_{new,1} \cup R^{-}_{1}) \bowtie \ldots \bowtie R^{-}_{j} \bowtie \ldots \bowtie (R_{new,n} \cup R^{-}_{n})) \cup \ldots \cup ((R_{new,1} \cup R^{-}_{1}) \bowtie \ldots \bowtie (R_{new,n-1} \cup R^{-}_{n-1}) \bowtie R^{-}_{n}) \cup (R_{old,1} \bowtie \ldots \bowtie R_{old,n-1} \bowtie R^{-}_{n})]$

where each R_n is a relation involved in the query Q, each $R_{old,n}$ is a state of the relation R_n as of the beginning of a preceding processing cycle, each $R_{new,n}$ is a state of the relation R_n as of the beginning of the current processing cycle, each R^{-}_{n} is a subset of tuples deleted from the relation R_n since the beginning of a preceding processing cycle, and each R^{+}_{n} is a subset of tuples added to the relation R_n since the beginning of a preceding processing cycle, and

if said results set is non-empty, designating the query Q as one of said identified queries.

25. The method claimed in claim 24, wherein said processing further comprises adjusting said results set to account for the over-invalidation term:

$\bowtie^{n}_{i} (R^{-}_{i} \cup R^{+}_{i}) - (\bowtie^{n}_{i} R^{-}_{i} \cup \bowtie^{n}_{i} R^{+}_{i})$

26. A system for producing dynamically generated content in response to user content requests that generate database queries, comprising:

a database management system;

a web server for receiving user requests for dynamically generated content and for providing said dynamically generated content; and

an application server coupled between the database management system and the web server for generating database polling queries corresponding to said requests and for supplying resulting data from the database to said web server,

the system performing an invalidation method to invalidate dynamically generated content generated by said system and stored in caches of network devices, said method comprising:

receiving updates to the database management system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing is performed using said database, which is prevented from being updated during said processing so as to reflect the state of the database at the beginning of said processing, and an update log that reflects all updates to the database since the beginning of the preceding processing cycle.

27. The system claimed in claim 26, wherein said processing comprises:

determining whether respective results sets for subsets of said set of queries are empty or non-empty for tuples added to and deleted from said database between the beginning of a preceding processing cycle and the beginning of the current processing cycle, using said locked database and said update log of said database; and

for each non-empty results set, designating the queries of the corresponding subset as ones of said identified queries.

28. The system claimed in claim 27, said processing further comprising, if a results set for a given query Q is non-empty, designating as identified queries one or more additional queries that are dependent from said query Q in a positive query lattice.

29. The system claimed in claim 27, said processing further comprising, if a results set for a given query Q is non-empty, skipping processing of one or more additional queries that are dependent from said query Q in a negative query lattice.

30. The system claimed in claim 26, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

31. The system claimed in claim 26, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

32. The system claimed in claim 26, wherein a subset of said queries comprises queries represented by a common query type and including a join operation, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, and to relations of said database utilized by queries of said query type for which there have been no updates since the preceding invalidation cycle, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

33. A system for producing dynamically generated content in response to user content requests that generate database queries, comprising:

a database management system;

a web server for receiving user requests for dynamically generated content and for providing said dynamically generated content; and

an application server coupled between the database management system and the web server for generating database polling queries corresponding to said requests and for supplying resulting data from the database to said web server,

the system performing an invalidation method to invalidate dynamically generated content generated by said system and stored in caches of network devices, said method comprising:

receiving updates to the database management system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing comprises, for a given query Q,

determining whether a results set is empty or non-empty, the results set consisting of the results:

$$[(R^{+}_{1} \bowtie R_{new,2} \bowtie \ldots \bowtie R_{new,n}) \cup \ldots \cup (R_{new,1} \bowtie \ldots \bowtie R^{+}_{j} \bowtie \ldots \bowtie R_{new,n}) \cup \ldots \cup (R_{new,1} \bowtie \ldots \bowtie R_{new,n-1} \bowtie R^{+}_{n})] \cup$$

$$[(R^{-}_{1} \bowtie (R_{new,2} \cup R^{-}_{2}) \bowtie \ldots \bowtie (R_{new,1} \cup R^{-}_{1})) \cup \ldots \cup ((R_{new,1} \cup R^{-}_{1}) \bowtie \ldots \bowtie R^{-}_{j} \bowtie \ldots \bowtie (R_{new,n} \cup R^{-}_{n})) \cup \ldots \cup ((R_{new,1} \cup R^{-}_{1}) \bowtie \ldots \bowtie (R_{new,n-1} \cup R^{-}_{n-1}) \bowtie R^{-}_{n}) \cup (R_{old,1} \bowtie \ldots \bowtie R_{old,n-1} \bowtie R^{-}_{n})]$$

where each R_n is a relation involved in the query Q, each $R_{old,n}$ is a state of the relation R_n as of the beginning of a preceding processing cycle, each $R_{new,n}$ is a state of the relation R_n as of the beginning of the current processing cycle, each R^-_n is a subset of tuples deleted from the relation R_n since the beginning of a preceding processing cycle, and each R^+_n is a subset of tuples added to the relation R_n since the beginning of a preceding processing cycle, and

if said results set is non-empty, designating the query Q as one of said identified queries.

34. The system claimed in claim 33, wherein said processing further comprises adjusting said results set to account for the over-invalidation term:

$$\bowtie^n_i (R^-_i \cup R^+_i) - (\bowtie^n_i R^-_i \cup \bowtie^n_i R^+_i)$$

35. In a system for producing dynamically generated content in response to user content requests that generate database queries, a method for invalidating dynamically generated content stored in caches of network devices, comprising:

receiving updates to a database of the system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing is performed using said database, which is freely updated during said processing, and a portion of an update log of said database that reflects updates to the database made prior to the beginning of said processing.

36. The method claimed in claim 35, wherein said processing comprises:

determining whether respective results sets for subsets of said set of queries are empty or non-empty for tuples added to and deleted from said database since the beginning of a preceding processing cycle, using said freely updated database and said locked update log of said database; and

for each non-empty results set, designating the queries of the corresponding subset as ones of said identified queries.

37. The method claimed in claim 36, further comprising, if a results set for a given query Q is non-empty, designating as identified queries one or more additional queries that are dependent from said query Q in a positive query lattice.

38. The method claimed in claim 36, further comprising, if a results set for a given query Q is non-empty, skipping processing of one or more additional queries that are dependent from said query Q in a negative query lattice.

39. The method claimed in claim 35, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

40. The method claimed in claim 35, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

41. The method claimed in claim 35, wherein a subset of said queries comprises queries represented by a common query type and including a join operation, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, and to relations of said database utilized by queries of said query type for which there have been no updates since the preceding invalidation cycle, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

42. The method claimed in claim 35, wherein said processing further comprises correcting an under-invalidation of a preceding processing cycle.

43. In a system for producing dynamically generated content in response to user content requests that generate database queries, a method for invalidating dynamically generated content stored in caches of network devices, comprising:

receiving updates to a database of the system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing comprises, for a given query Q,

determining whether a results set is empty or non-empty, the results set consisting of the results:

$$\begin{aligned}
&(R^{+}_{1} \bowtie (R'_{2} \cup R^{+}_{2}) \bowtie \ldots \bowtie (R'_{n} \cup R^{+}_{n})) \cup \ldots \\
&\quad \cup ((R'_{1} \cup R^{+}_{1}) \bowtie \ldots \bowtie R^{+}_{j} \bowtie \ldots \bowtie (R'_{n} \cup R^{+}_{n})) \cup \ldots \\
&\quad \cup ((R'_{1} \cup R^{+}_{1}) \bowtie \ldots \bowtie (R'_{n-1} \cup R^{+}_{n-1}) \bowtie R^{+}_{n}) \cup \\
&(R^{-}_{1} \bowtie (R'_{2} \cup R^{-}_{2}) \bowtie \ldots \bowtie (R'_{n} \cup R^{-}_{n})) \cup \ldots \\
&\quad \cup ((R'_{1} \cup R^{-}_{1}) \bowtie \ldots \bowtie R^{-}_{j} \bowtie \ldots \bowtie (R'_{n} \cup R^{-}_{n})) \cup \ldots \\
&\quad \cup ((R'_{1} \cup R^{-}_{1}) \bowtie \ldots \bowtie (R'_{n-1} \cup R^{-}_{n-1}) \bowtie R^{-}_{n})
\end{aligned}$$

where each R_n is a relation involved in the query Q, each R'_n is a state of the relation R_n as of the time of processing of the query Q, each R^{-}_{n} is a subset of tuples deleted from the relation R_n since the beginning of a preceding processing cycle, and each R^{+}_{n} is a subset of tuples added to the relation R_n since the beginning of a preceding processing cycle, and

if said results set is non-empty, designating the query Q as one of said identified queries.

44. A system for producing dynamically generated content in response to user content requests that generate database queries, comprising:

a database management system;

a web server for receiving user requests for dynamically generated content and for providing said dynamically generated content; and an application server coupled between the database management system and the web server for generating database polling queries corresponding to said requests and for supplying resulting data from the database to said web server,

the system performing an invalidation method to invalidate dynamically generated content generated by said system and stored in caches of network devices, said method comprising:

receiving updates to the database management system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing is performed using said database, which is freely updated during said processing, and a portion of an update log of said database that reflects updates to the database made prior to the beginning of said processing.

45. The system claimed in claim 44, wherein said processing comprises:

determining whether respective results sets for subsets of said set of queries are empty or non-empty for tuples added to and deleted from said database since the beginning of a preceding processing cycle, using said freely updated database and said locked update log of said database; and

for each non-empty results set, designating the queries of the corresponding subset as ones of said identified queries.

46. The system claimed in claim 45, further comprising, if a results set for a given query Q is non-empty, designating as identified queries one or more additional queries that are dependent from said query Q in a positive query lattice.

47. The system claimed in claim 45, further comprising, if a results set for a given query Q is non-empty, skipping processing of one or more additional queries that are dependent from said query Q in a negative query lattice.

48. The system claimed in claim 44, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

49. The system claimed in claim 44, wherein a subset of said queries comprises queries represented by a common query type, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

50. The system claimed in claim 44, wherein a subset of said queries comprises queries represented by a common query type and including a join operation, and

wherein processing of said subset comprises:

issuing polling queries to a query instance table of said query type, and to update tables constructed from said update log and corresponding to relations utilized by queries of said query type, and to relations of said database utilized by queries of said query type for which there have been no updates since the preceding invalidation cycle, for each of said updates; and

designating query instances identified through said polling queries as said identified queries.

51. The system claimed in claim 44, wherein said processing further comprises correcting an under-invalidation of a preceding processing cycle.

52. A system for producing dynamically generated content in response to user content requests that generate database queries, comprising:

a database management system;

a web server for receiving user requests for dynamically generated content and for providing said dynamically generated content; and

an application server coupled between the database management system and the web server for generating database polling queries corresponding to said requests and for supplying resulting data from the database to said web server,

the system performing an invalidation method to invalidate dynamically generated content generated by said system and stored in caches of network devices, said method comprising:

receiving updates to the database management system;

periodically performing processing on a set of received queries to identify queries for which corresponding dynamically generated content will be invalidated as a result of said updates; and

sending invalidation messages to devices storing said corresponding content of said identified queries,

wherein said processing comprises, for a given query Q,

determining whether a results set is empty or non-empty, the results set consisting of the results:

$$
\begin{aligned}
&(R^{+}_{1} \bowtie (R'_{2} \cup R^{+}_{2}) \bowtie \ldots \bowtie (R'_{n} \cup R^{+}_{n})) \cup \ldots \\
&\quad \cup ((R'_{1} \cup R^{+}_{1}) \bowtie \ldots \bowtie R^{+}_{j} \bowtie \ldots \bowtie (R'_{n} \cup R^{+}_{n})) \cup \ldots \\
&\quad \cup ((R'_{1} \cup R^{+}_{1}) \bowtie \ldots \bowtie (R'_{n-1} \cup R^{+}_{n-1}) \bowtie R^{+}_{n}) \cup \\
&(R^{-}_{1} \bowtie (R'_{2} \cup R^{-}_{2}) \bowtie \ldots \bowtie (R'_{n} \cup R^{-}_{n})) \cup \ldots \\
&\quad \cup ((R'_{1} \cup R^{-}_{1}) \bowtie \ldots \bowtie R^{-}_{j} \bowtie \ldots \bowtie (R'_{n} \cup R^{-}_{n})) \cup \ldots \\
&\quad \cup ((R'_{1} \cup R^{-}_{1}) \bowtie \ldots \bowtie (R'_{n-1} \cup R^{-}_{n-1}) \bowtie R^{-}_{n})
\end{aligned}
$$

where each R_n is a relation involved in the query Q, each R'_n is a state of the relation R_n as of the time of processing of the query Q, each R^{-}_{n} is a subset of tuples deleted from the relation R_n since the beginning of a preceding processing cycle, and each R^{+}_{n} is a subset of tuples added to the relation R_n since the beginning of a preceding processing cycle, and

if said results set is non-empty, designating the query Q as one of said identified queries.